УДК 338.512:338.583:338.585

© Прихнич А.М.
магістр,
© Гальчинський Л.Ю.
канд. техн. наук, доцент

ОЦІНКА ВАРТОСТІ РОЗРОБКИ ПРОГРАМНИХ ДОДАТКІВ

Вступ. Сьогодні розробка програмних засобів (ПЗ) є одним з досить прибуткових видів бізнесу, який не вимагає великих капіталовкладень в основні фонди. Однак він поставив ряд цікавих задач, які й досі до кінця не були вирішені. Саме до них належить і задача оцінки вартості розробки програмного забезпечення.

Програмні проекти часто являються непередбачуваними, а їх область дії часто не може бути визначеною. При розробці ПЗ, на відміну від багатьох інших галузей промисловості, один і той самий продукт ніколи не буде створюватись двічі [1]. На додачу до оцінок, які враховують відмінності між специфікаціями нового та попереднього проектів, також необхідно враховувати відмінності в середовищі розробки та доставки, приймаючи до уваги швидкі пооперацiйні зміни в технології. При цьому можуть не використовуватись хронологічні дані до переходу в нове середовище. Оцінка розмірів ПЗ і трудовитрат, що пов'язані з розробкою, може виконуватись багато разів протягом всього життєвого циклу. І у кожному наступному разі зростає ступінь достовірності оцінки.

Постановка задачі. Для оцінки розробки програмного забезпечення менеджмент вимагає використовувати кількісні оцінки та моделі оцінювання, які базуються на теорії та збирають статистичні дані минулих проектів. Багато моделей було розроблено в останні 20 років, які прогнозують вартість програмного забезпечення та визначають графіки розробки на ранніх стадіях створення продуктів, наприклад, одразу після визначення вимог до програмних засобів. Всі стадії розробки програмного забезпечення показані на рис. 1.

Ці оцінки управляють процесом розробки в розрахунку розробити продукт вчасно, в рамках бюджету та з очікуваним рівнем якості. Не дивлячись на надмірну кількість моделей, більшість проектів по розробці програмного забезпечення мають довший графік та перевитрату коштів. Одна з причин цього полягає у тому, що більшість з існуючих моделей не повні та не заповнені точними оцінками. Із зростанням відношення вартості програмного забезпечення до вартості деталей [2], акценти перемістились на покращення точності оцінки вартості розробки програмних продуктів.

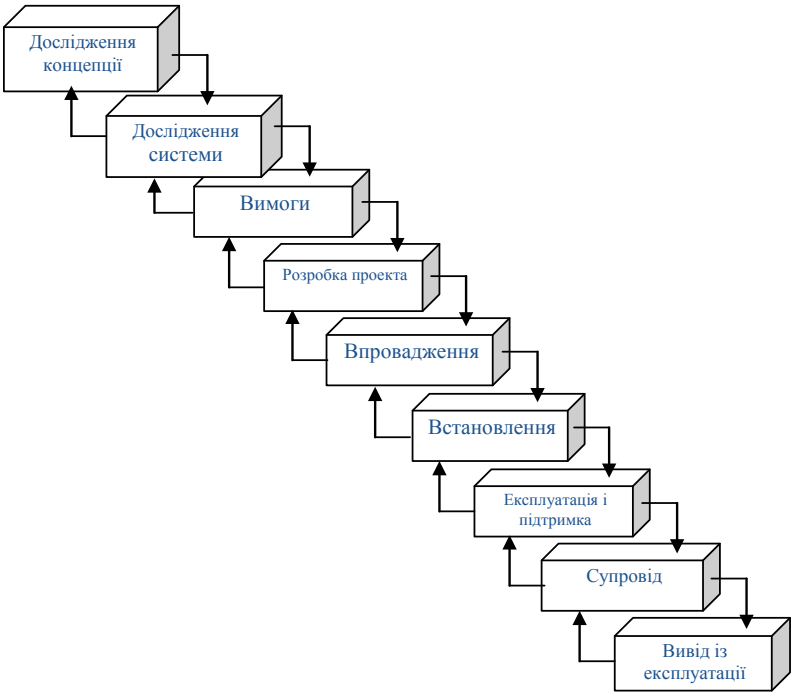

Рис. 1. Стадії розробки програмних засобів

Таким чином перед нами постала задача збільшення точності прогнозів, яку ми плануємо вирішити шляхом удосконалення та уточнення існуючої моделі COCOMO II.

Методологія. При дослідженні ми використовували наступну методологічну базу: регресійний аналіз; порівняльний аналіз; модель конструктивних витрат; нейро мережі.

Регресійний та порівняльний аналізи ми використовували під час аналізу статистичних даних для виявлення певних залежностей та закономірностей. За основу наших досліджень ми взяли модель

конструктивних витрат (COnstructive COst MOdel II, COCOMO II) [3]. Однак дана модель не дала нам необхідної точності розрахунків і ми використали нейро мережі для підвищення точності вхідних параметрів і, як результат, підвищили точність остаточних вихідних оцінок.

Результати дослідження. За основу наших обчислень ми взяли модель COCOMO II, яка є поліпшеною версією вихідної моделі COCOMO. Заснована на використанні регресії, модель була розроблена доктором Барі В. Боемом (Dr. Barry W. Boehm) на початку 1970 років.

Модель COCOMO II виглядає наступним чином:

$$E = a \cdot (розмір)^{b} \cdot \prod_{i=1}^{15} \mathrm{EAF}_i \qquad (1)$$

де a і b - константи, визначені на етапі регресійного аналізу (в залежності від проекту).

Розмір - тисячі рядків коду (KLOC).

E - трудовитрати, виражені в людино-місяцях

EAF – драйвер витрат або фактор коректування трудовитрат (Effort adjustment factor)

Завдяки цій моделі полегшується виконання оцінки для об’єктно-орієнтованого ПЗ, програм, створених із застосуванням спіральної або еволюційної моделей, а також додатків, розроблених на базі готових комерційних програм. Зокрема, ця модель застосовується при оцінці програмних витрат на розробку баз даних і можливостей підтримки інструментів у процесі безперервного поліпшення моделі [4]. Ця модель також формує аналітичну основу, набір інструментів і технік, застосовуваних при оцінці ефектів, зв'язаних з поліпшенням технології розробки ПЗ на базі графіків і витрат на фазі життєвого циклу розробки ПЗ.

У композиційній прикладній моделі COCOMO II для виконання оцінок застосовується метод об'єктних точок. При використанні цієї моделі передбачається використання інтегрованих CASE-інструментів з метою виконання швидкого прототипування. Об'єкти включають екрани, звіти і модулі, що мають відношення до мов програмування третього покоління. Оцінюється кількість фізичних об'єктів, складність кожного об'єкта, а також обчислюється зважений підсумок (підраховується кількість об'єктних точок) [2]. Також оцінюється процентне співвідношення показників повторного використання й очікуваної продуктивності. На основі цієї інформації може виконуватися оцінка трудовитрат.

У моделі COCOMO II явно забезпечується доступ до додаткової інформації на пізніх стадіях проекту, обробляються нелінійні витрати при повторному використанні програмних компонентів, а також оцінюються ефекти впливу декількох факторів по шкалі показників збитків [5]. Деякі з перерахованих параметрів являють собою оцінку обороту персоналу, географічний розподіл команди, а також "зрілість" процесу розробки в тому виді, у якому вона описується Інститутом SEI.

Основним недоліком моделей COCOMO та COCOMO II є їх недостатня точність та потреба коригувати драйвери затрат. Причина цього недоліку коріниться в природі регресійного підходу – вибором функціональної залежності ми постулюємо закон витрат, який наперед невідомий [6]. Відтак на точність впливає два фактори – адекватність гіпотези про функціональну залежність і якість даних.

Для підвищення точності отримуваних результатів нам необхідно підвищити точність драйверів затрат. Це є складною задачею оскільки кожен із 15 драйверів затрат буде специфічним для кожного нового проекту [7]. Для вирішення цієї задачі ми побудували нейро мережі, які на основі початкових вагових коефіцієнтів видають драйвери затрат, характерні саме для даного конкретного проекту і фактично самі визначають тип функціональної залежності. Розрахунки показали, що моделі, які базуються на застосуванні нейромереж, мають кращу якість прогнозу, ніж деякі стастичні моделі, при оцінюванні достовірності прогнозу та кількості помилок. Ми дослідили додаток головних компонент аналізу в моделюванні нейромереж як спосіб підвищити якість прогнозу моделей нейромереж.

Висновки. У даній роботі ми вирішили проблему покращення точності оцінок вартості розробки ПЗ шляхом побудови нейро мереж. Кожна з побудованих мереж дозволяє отримати драйвер витрат характерний саме для даного проекту. Причому якщо добуток усіх 15 драйверів затрат менше 1, то він знизить вартість, якщо більше – то збільшить, а якщо рівний 1, то вартість розробки лишиться незмінною. Таким чином, наші нейро мережі дозволяють вирішити проблему покращення точності оцінки вартості програмного забезпечення, незалежно від його специфіки.

Література:

1. Шафер Д. Ф., Фатрелл Р. Т., Шафер Л. И. Управление программными проектами: достижение оптимального качества при минимуме затрат. – М.: Вильямс, 2004. – 1136 с.: ил.
2. Chulani S. Bayesian analysis of empirical software engineering cost models // Series on software engineering and knowledge engineering, 2005 – pp. 41-51.
3. Gray A. R., MacDonnell S. G. A Comparison of techniques for developing predictive models for software metrics // Information and software technology. –1997. – № 39. – pp. 45-50.
4. Karunanithi N., Whitley D., Malaiya Y.K. Using neural networks in reliability prediction // IEEE Software. – 1992. № 4. – pp. 53-59.
5. Khoshgoftaar T.M., Pandya A.S., Lanning D.L. Application of neural networks for predicting program faults // Annals of software engineering.–1995. – № 1. – pp. 141-154.
6. Lyu M.R. Handbook of software seliability engineering // IEEE Computer Society Press. – 1996. – № 3. – pp.

219-254.
7. Poulin J.S., Wesley A. Measuring software reuse, principles, practices and economic models. – Boston.: Addison-Wesley Longman. 1997. – 195 p.

Рисунок 1